IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 3, September 2024, pp. 3634~3643
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i3.pp3634-3643

Comparison and evaluation of YOLO models for vehicle detection on bicycle paths

Joshue Garcia-Pajuelo, Ernesto Paiva-Peredo
Faculty of Systems Engineering and Electronics, Electronic Engineering, Universidad Tecnológica del Perú, Lima, Peru

Article Info

Article history:

Received Jan 28, 2024
Revised Feb 12, 2024
Accepted Feb 28, 2024

Keywords:

Bicycle
Bicycle lanes
Deep learning
Motorcycle
Vehicle detection
YOLO
YOLOv8

ABSTRACT

Non-permitted vehicles have taken over bicycle lanes in various Latin American cities as an alternative escape from traffic. Still, they do not foresee the risk to which they expose users of smaller vehicles, such as cyclists. Technological advancement has made researchers use deep learning (DL) to solve various problems in a city's traffic. However, no research has been found focusing on any issue of vehicles allowed or prohibited to travel on a bicycle lane. Therefore, in this article, the you only look once (YOLO) algorithm was used, taking the lightest models from the YOLOv4 to the most recent version, YOLOv8, to detect 05 classes of vehicles that transit or interfere in a bicycle lane, such as bicycles (Bi), motorcycles (Mo), electric motorcycles (ME), electric scooters (SE), and motorcycle cabs (Mt). When testing with the test images, the YOLOv8m model in 50 epochs, using a batch size of 32 and stochastic gradient descent (SGD) optimizer, was the most optimal, obtaining F1 results with 88.00%, mAP@0:50 of 94.80% and mAP@0.50:0.95 of 76.60%, also had a training time of 1:28h using a Nvidia T4 GPU from Google Colab.

Corresponding Author:

Ernesto Paiva-Peredo
Faculty of Systems Engineering and Electronics, Electronic Engineering, Universidad Tecnológica del Perú
Jirón Hernán Velarde 179, Lima 15046, Lima, Peru
Email: c22395@utp.edu.pe

1. INTRODUCTION

The significant vehicular congestion in several cities in Latin America has led to various environmental problems [1] and, with the subsequent arrival of the pandemic, has generated modifications or changes in the habits of many people, so many of them chose to use a bicycle daily. Likewise, the authorities in this part of the continent, under all this context, took measures to increase the use of bicycles and implement more bicycle lanes since the bicycles were not only used recreationally but also as a means of transportation to avoid traffic congestion in the transfer to work centers or homes. For this reason, this research addresses a crucial problem in several Latin American cities: the invasion of bicycle lanes by unpermitted vehicles, representing a significant risk for users of smaller vehicles, such as cyclists. This phenomenon affects the mobility and safety of citizens and contributes to traffic disorder. Advanced technologies, such as deep learning (DL) through YOLO algorithms, are presented as an innovative solution to address this problem, specifically in vehicle detection on bicycle lanes. By focusing on this problem, our research aims to improve road safety on bicycle lanes and contribute to traffic efficiency.

With the help of technology, we try to order traffic [2]–[5], seeking to comply with traffic signs and rules [6], as well as to avoid accidents caused by any type of vehicle [7]–[10]. For this reason, to improve traffic problems in cities, they have tried to use in [11] convolutional neuronal networks (CNN) AlexNet algorithm [12] to recognize cars, trucks, buses, motorcycles, and vans. However, it did not effectively recognize the different types of vehicles due to the similarities of their characteristics, which is why they modified the algorithm calling it ProAlexNet obtaining an accuracy of 95.6%, 10.1% above the AlexNet algorithm. Also, Fanthony *et al.* [13] implemented an autonomous vehicle that can identify people, motorcycles, and cars at different distances, making use of algorithms such as k-nearest neighbors (KNN) [14] and mask region convolutional neural network (R-CNN) [15] to detect objects. However, they have a good recognition accuracy; they cannot be executed in real-time as the YOLO algorithm [16] did, so they came to use YOLOv4 [17], obtaining 83% accuracy in its implementation.

Different items have been proposed in the Philippines to improve their transport system. In one of the investigations [18], they joined known CNNs forming Faster R-CNN inception [19], [20] and single shot multibox detector (SSD) MobilNetV2 [21], [22]. Concluding, Faster R-CNN Inception had a high accuracy of 86.40%, but in their test videos, there was 75.00% more delay than the original, and consequently, the detection was slow. On the other hand, using SSD MobilNetV2 maintained the speed of the test video but had a lower accuracy of 78.77%. However, in investigations in several cities around the world, good results were obtained using the YOLO algorithm [23]–[26], on this occasion [27], used YOLOv3 [28], to detect objects with similar characteristics such as a bicycle and a motorcycle, in addition to being able to recognize them in different scenarios such as morning, noon, afternoon, evening, night and rainy. In the case of motorcycles, an average performance of 55.70%, 49.71%, 26.30%, 42.18%, and 56.51%, respectively, was obtained. Meanwhile, in the case of bicycles, an average yield of 42.38%, 66.58%, 43.42%, and 43.8%, respectively, was obtained. The conclusion is that the performance improves as the model is given a greater variety of images from different perspectives.

Miao *et al.* [29] found it inconvenient for vehicle detection at night using algorithms such as Faster R-CNN and SSD, but using YOLOv3, they found a better performance, obtaining an average accuracy of 93.66%, being 6.14% higher than Faster R-CNN and 3.21% higher than SSD. In addition, classical CNNs [30]–[32] have the problem of consuming more computational resources that do not make them suitable for real-time detection. Also, Wu *et al.* [33], a system using YOLOv3 was implemented to detect and count cars, heavy vehicles, and motorcycles at highway intersections. In this way, they could have better control and order of traffic, taking advantage of the video surveillance cameras installed on the roads and obtaining 98.00% accuracy.

In searching for better real-time detection, there are works like [34], which focus on making comparisons between algorithms to decide on the best option, so it took YOLOv4, YOLOv4-Tiny [35], YOLOv5s from which average recognition accuracy was obtained 96.97%, 97.45%, and 95.1%, respectively. On the other hand, the processing times to detect an image were 32.953 ms, 15.546 ms, and 28.00 ms, respectively. They concluded that YOLOv4-Tiny gives you better recognition accuracy and processing time to detect an object. In the other part, Gomes *et al.* [36], a people and bicycle counter was performed using Jetson Nano hardware; therefore, they required software light enough but efficient enough that can run in real-time through a device with limited resources, concluding that the best results are given with YOLOv5n and YOLOv5s which had an average accuracy 49.25% and 60.51%, respectively. The processing time to detect was 66.99ms and 100.06ms, respectively.

Based on the research collected, it is of utmost importance for different parts of the world to have orderly transportation where vehicles comply with traffic regulations to avoid accidents. Therefore, using DL algorithms to detect objects circulating on roads, streets, and highways brings a great benefit because they can detect any vehicle or object. In addition, when detection in real-time is required, the tendency is to use YOLO algorithms due to their single-stage architecture, which performs the processes in a single network. Likewise, the projects can be molded among its various versions' models, obtaining satisfactory results concerning other algorithms. However, it is observed that the investigations do not cover the bicycle lanes nor the problems that exist in these, which are also part of any city's transportation and should have the same priority. Therefore, this research will detect bicycles, motorcycles, electric motorcycles, electric scooters, and motorcycle cabs using various models of YOLO.

2. METHOD

2.1. Dataset

The images were collected from the results of an internet search engine and images from social networks related to the research. Additionally, the selection of images attempted to meet different conditions, considering image size, type of lighting, situations, and backgrounds. Then, five vehicle classes transiting or interfering in a bicycle lane were identified; the classes were 0, 1, 2, 3, and 4, labeled with bicycles in Figure 1(a), electric motorcycles in Figure 1(b), motorcycles in Figure 1(c), motorcycle cabs in Figure 1(d), and electric scooters in Figure 1(e), respectively, more details can be found in https://www.dropbox.com/scl/fo/ervqpcgd20tyxi322plwl/h?rlkey=uga7uy9mtog6w2yr37tskad1r&dl=0). Figure 1 shows some images collected for the research where the characteristics considered can be appreciated. Finally, the images were filtered with the naked eye using the software Find.Same.Images.OK, and duplicate cleaner pro.

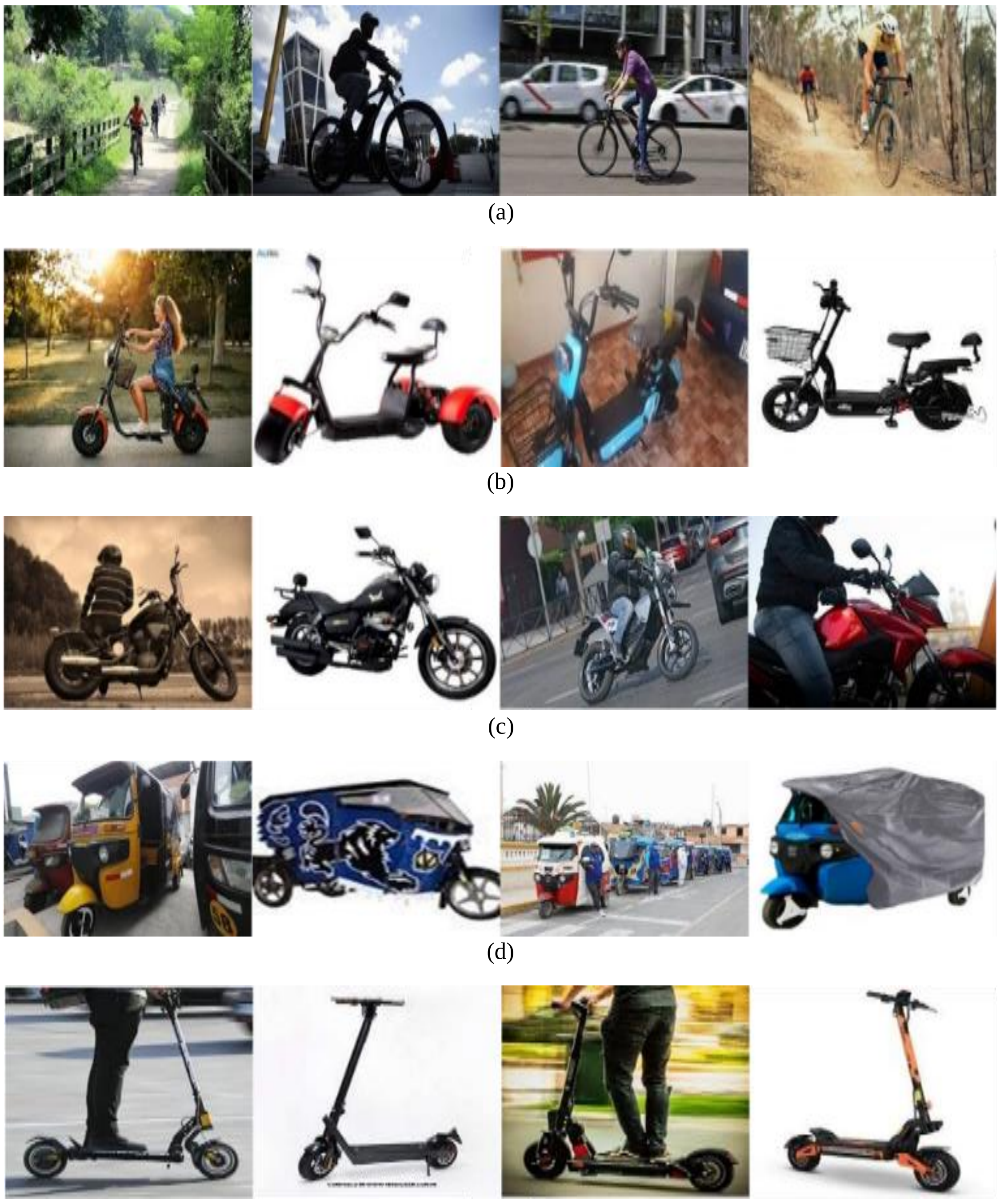


Figure 1. Examples of images used in the investigation are (a) bicycles, (b) electric motorcycles, (c) motorcycles, (d) motorcycle cabs, and (e) electric scooters

Table 1 shows that the database comprises 4852 images, divided into 1640 images of bicycles, 535 of electric motorcycles, 1362 of motorcycles, 617 of motorcycle cabs, and 698 of electric scooters. The investigation had to be segmented into three groups: approximately 60% was taken for training, 20% for validation, and 20% for testing, obtaining 2915 training images, 973 images for model validation, and 964 images for testing. In addition, in Table 2, a total of 6611 instances are displayed; there were 4071 for training, 1296 for validation, and 1244 for testing.

Table 1. Segmentation of the number of images

Data Classes	Bicycle	Moto electric	Motorcycle	Mototaxi	Scooter electric	Total
Train	986	321	818	371	419	2915
Val	329	107	273	124	140	973
Test	325	107	271	122	139	964
Total	1640	535	1362	617	698	4852

Table 2. Segmentation of the number of instances

Data Classes	Bicycle	Moto electric	Motorcycle	Mototaxi	Scooter electric	Total
Train	1547	345	1119	578	482	4071
Val	528	112	328	164	164	1296
Test	450	117	362	149	166	1244
Total	2525	574	1809	891	812	6611

2.2. Architecture of versions YOLO

The YOLO algorithm was released in 2015 [37] and is used for real-time object detection using a single network, with convolutional layers and fully connected layers, which are distributed in three components: backbone, in charge of extracting all types of features in an image; neck, collects the extracted features; and head, generates the bounding box coordinates for prediction and detection. Recent versions have greatly improved inference speed, model size reduction, and accuracy. By 2020, YOLOv4 [17] was released, changing the network in each of its components and implementing functional improvements of a lighter version as YOLOv4-tiny [35], which reduces the convolutional layers in its backbone and the number of YOLO layers in its Head. In the same year, Ultralytics launched YOLOv5 using PyTorch as an environment, including different models from very light to a heavier one, but with higher accuracy. Additionally, Ultralytics incorporated the focus layer, which reduces the graphics processing unit (GPU) requirements.

In 2022, the Chinese company Meituan launched YOLOv6 [38], improving the speed of inference from anchor-free bounding boxes. The same year, YOLOv7 [39] appeared, completely changing the backbone's (E-ELAN) architecture to improve the learning capacity and be more effective when training. By 2023, Ultralytics launched YOLOv8, taking free anchor features, avoiding the degradation of the trained model, and creating a much simpler API.

In Table 3, we have compiled some data to be considered for the models used, such as image size, model size, parameters, and floating-point operations per second (FLOPs). It can be seen that the heaviest and most computationally demanding models are YOLOv4 and YOLOv7, while the lightest and least computationally consuming models are YOLOv5n and YOLOv6Lite-L. In the research, we tried to use the lightest models of each version according to the pre-training tables given by their authors, taking from YOLOv4 onwards.

Table 3. YOLO architectures data

Models	Image size (pixel)	Model size (MB)	Parameters (M)	FLOPs (G)
YOLOv4-tiny	416	23	5.88	6.8
YOLOv4	416	250.10	61.6	59.7
YOLOv5n	640	3.74	1.9	4.5
YOLOv5m	640	41.17	21.2	49
YOLOv6Lite-L	320	2.48	1.09	0.87
YOLOv6n	640	10.21	4.7	11.4
YOLOv7-tiny	640	11.98	6.05	13.2
YOLOv7	640	73.06	37.22	105.2
YOLOv8n	640	6.07	3.2	8.7
YOLOv8m	640	50.78	25.9	78.9

The hyperparameters modified and adjusted during training are: batch size, taking batches of 16 and 32, for the case of YOLOv4-tiny, a batch of 64 was added; optimizer, for the versions that allow it, the stochastic gradient descent (SGD) and Adam algorithms were taken; and epoch, evaluating the models in 50, 100 and 200. The performance was measured through the metrics delivered by each model; in this way, it was assessed if it was trained correctly or if the results obtained were ideal. The metrics used in the research were F1-score, mean average precision (mAP), training time, and confusion matrix for the best model obtained. Colab and Nvidia T4 GPU with the free version were used as training environments. But, for those YOLO models with many parameters and FLOPs, the paid version (Colab pro) was required, using the Nvidia V100 GPU.

To select the best model, we first took the highest F1-score and mAP values of all the models obtained during training. Then, the models with high training times with similar or lower values in the F1-score and mAP metrics were discarded. Afterward, each metric was evaluated, starting with the F1-score, where the models with values well below the others were discarded; when models with similar values were left, the metric mAP@0.50 was evaluated and then, mAP@0.50:0.95, where those models with values below the rest were discarded, thus obtaining the model with the most suitable values.

3. RESULTS AND DISCUSSION

The results of YOLOv4 and YOLOv4-tiny were evaluated separately since the training for these models varies concerning the later versions. According to their specifications, each class must have at least 2000 iterations to obtain correct training. From YOLOv5 onwards, training and validation are similar for the various versions. In the case of the YOLOv7 model and version, no results could be obtained when a batch size of 32 was used with any optimizers; the T4 and V100 GPUs were insufficient for training due to the high computational consumption. All models were evaluated with the Test images.

In Table 4 shows that the best classification performance (F1) with 90% is of the YOLOv4 model with batch size 32, compared to 85.00% of YOLOv4-tiny with batch size 64. On the other hand, when evaluating the detection performance (mAP), it can be observed that YOLOv4-tiny with batch size 32 achieves 90.13%. We can conclude that the YOLOv4 model with batch size 32 is above the rest because, for F1, it greatly exceeds the other models, while in mAP, it is only below 0.3%, concerning the maximum value achieved. In addition, for YOLOv4, a V100 GPU was used due to its high computational consumption, obtaining very high training times compared to YOLOv4-tiny, where the training times of its models were much lower using a less powerful GPU such as the T4.

Table 4. Results of YOLOv4 and YOLOv4-tiny

Model	Iteration	Batch	F1-score (%)	mAP@0.50 (%)	Time to train (h)	GPU
YOLOv4-tiny	10k iterations	16	67.00	75.06	00:45	T4
		32	83.00	**90.13**	01:28	T4
		64	85.00	89.71	02:32	T4
YOLOv4	10k iterations	16	88.00	87.84	01:54	V100
		32	**90.00**	89.83	03:38	V100

In Table 5, the best F1 value is 89.00% of the models YOLOv5n in 200 epochs, YOLOv5m in 50 epochs and YOLOv8m in 100 epochs. Evaluating mAP@0.50, the best result was YOLOv6n in 200 epochs with 94.00%. Meanwhile, the best value mAP@0.50:0.95 is 74.60% of the YOLOv8m models in 50 and 100 epochs. It can be concluded that the YOLOv8m model in 100 epochs is the most acceptable of all since when comparing metrics such as F1 and mAP@0.50:0.95, it has the highest values, and in mAP@0.50, it is below by only 0.10, % concerning the maximum value obtained. However, the training time of 2:55h with the T4 GPU does not make it ideal since there are other options with similar performances in less time.

Table 5. Model results using SGD optimizer with batch size 16

Model	Epoch	F1-score (%)	mAP@0.50 (%)	mAP@0.50:0.95 (%)	Time to train (h)	GPU
YOLOv5n	50	88.00	92.00	66.60	00:33	T4
	100	88.00	92.50	68.20	01:31	T4
	200	**89.00**	92.40	69.50	03:00	T4
YOLOv5m	50	**89.00**	93.70	72.10	01:13	T4
	100	88.00	91.90	72.20	02:30	T4
	200	88.00	91.70	70.40	01:52	V100
YOLOv6Lite-L	50	84.20	91.70	70.10	01:38	T4
	100	84.50	92.10	71.10	03:32	T4
	200	82.90	91.40	71.20	02:23	V100
YOLOv6n	50	85.50	93.00	71.70	01:59	T4
	100	83.00	91.90	71.40	03:50	T4
	200	87.80	**94.00**	72.90	02:25	V100
YOLOv7-tiny	50	85.00	90.40	64.30	00:54	T4
	100	85.00	90.30	63.60	01:47	T4
	200	86.00	90.10	65.40	01:55	V100
YOLOv7	50	84.00	88.70	64.70	02:42	T4
	100	86.00	91.20	67.30	01:58	V100
	200	88.00	92.30	70.70	03:55	V100
YOLOv8n	50	87.00	91.80	73.10	00:58	T4
	100	87.00	91.20	72.00	01:56	T4
	200	85.00	90.40	71.00	01:12	V100
YOLOv8m	50	88.00	92.70	**74.60**	01:27	T4
	100	**89.00**	93.90	**74.60**	02:55	T4
	200	87.00	91.60	71.90	02:00	V100

In Table 6, the best F1 and mAP@0.50 values were from the YOLOv6n model in 100 epochs, obtaining 89.50% and 94.60%, respectively. For mAP@0.50:0.95, the best value was 74.40% of the YOLOv8m model in 50 epochs. It is concluded that the YOLOv6n model in 100 epochs is the most acceptable of all since it has the best performances, except in the metric mAP@0.50:0.95, where it is below 1.40% concerning the best value. However, the training time was 3:43h using a T4 GPU, which is unsuitable as there are other lower times with similar performances.

Table 6. Model results using Adam optimizer with Batch size 16

Model	Epoch	F1-score (%)	mAP@0.50 (%)	mAP@0.50:0.95 (%)	Time to train (h)	GPU
YOLOv5n	50	86.00	90.30	64.30	00:44	T4
	100	87.00	90.80	67.10	01:27	T4
	200	86.00	90.00	67.80	02:41	T4
YOLOv5m	50	88.00	92.30	69.80	01:19	T4
	100	87.00	91.20	70.00	02:35	T4
	200	88.00	91.70	70.60	01:56	V100
YOLOv6Lite-L	50	84.10	91.80	71.60	01:28	T4
	100	83.60	91.40	71.60	03:11	T4
	200	82.70	89.90	70.10	02:10	V100
YOLOv6n	50	84.40	92.10	70.60	01:44	T4
	100	**89.50**	**94.60**	73.00	03:43	T4
	200	85.00	92.50	71.40	01:57	V100
YOLOv7-tiny	50	83.00	89.30	61.30	01:10	T4
	100	86.00	90.90	63.80	02:10	T4
	200	86.00	91.30	66.30	02:00	V100
YOLOv7	50	79.00	84.20	57.10	02:13	T4
	100	83.00	88.70	61.50	02:00	V100
	200	86.00	90.50	65.70	03:56	V100
YOLOv8n	50	88.00	92.40	73.70	01:00	T4
	100	87.00	92.80	73.60	02:00	T4
	200	86.00	89.70	70.20	01:10	V100
YOLOv8m	50	88.00	93.60	**74.40**	01:31	T4
	100	86.00	92.00	72.30	02:58	T4
	200	89.00	93.10	72.60	02:10	V100

In Table 7, the best F1 value is 90.00% of the YOLOv5m model in 50 and 100 epochs. Evaluating mAP@0.50 and mAP@0.50:0.95, the best results were of YOLOv8m in 50 epochs with 94.80% and 76.60%, respectively. It can be concluded that the YOLOv8m model in 50 epochs is the most optimal of all since when comparing metrics such as mAP@0.50 and mAP@0.50:0.95, it has the highest values, and F1 is below by 2.00% concerning the maximum value obtained. In addition, the training time of 1:28h using a T4 GPU is acceptable.

Table 7. Model results using SGD optimizer with Batch size 32

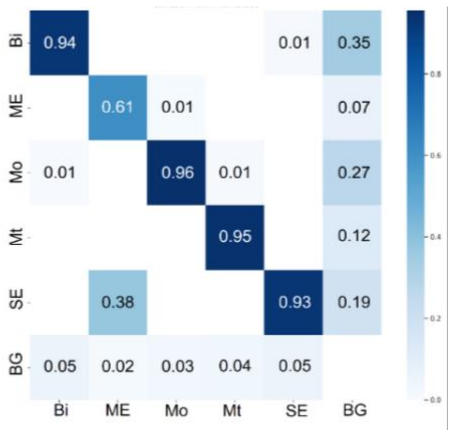
Model	Epoch	F1-score (%)	mAP@0.50 (%)	mAP@0.50:0.95 (%)	Time to train (h)	GPU
YOLOv5n	50	89.00	92.10	67.60	00:36	T4
	100	87.00	91.60	68.00	01:21	T4
	200	87.00	90.70	68.20	03:15	T4
YOLOv5m	50	**90.00**	93.30	72.60	01:16	T4
	100	**90.00**	93.50	72.60	02:33	T4
	200	87.00	91.50	70.80	01:43	V100
YOLOv6Lite-L	50	83.40	90.50	69.20	01:38	T4
	100	83.40	91.50	70.50	03:12	T4
	200	83.20	91.70	71.20	02:16	V100
YOLOv6n	50	85.50	93.60	72.40	01:59	T4
	100	83.90	92.60	71.90	03:31	T4
	200	84.00	92.60	71.80	01:49	V100
YOLOv7-tiny	50	84.00	89.90	63.40	01:10	T4
	100	88.00	93.30	69.30	01:59	T4
	200	87.00	91.30	66.00	01:56	V100
YOLOv8n	50	88.00	93.30	74.00	01:00	T4
	100	87.00	93.10	74.20	01:55	T4
	200	88.00	94.20	74.10	00:58	V100
YOLOv8m	50	88.00	**94.80**	**76.60**	01:28	T4
	100	87.00	92.60	74.10	02:52	T4
	200	88.00	93.80	74.60	01:56	V100

In Table 8, the best F1 value was 90.00% of the YOLOv8m model in 200 epochs. For mAP@0.50, the best result was YOLOv8m in 50 and 200 epochs, with 94.00%. Meanwhile, the best value mAP@0.50:0.95 is 75.60% of the YOLOv8m model in 50 epochs. It is concluded that the YOLOv8m model in 50 epochs is the most optimal of all since when comparing metrics such as mAP@0.50 and mAP@0.50:0.95, it has the highest values, and F1 is below by 1.00% concerning the maximum value obtained. Furthermore, the training time of 1:26h is acceptable using a T4 GPU.

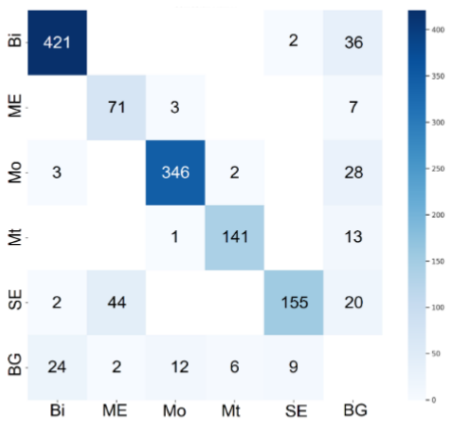
Table 8. Model results using Adam optimizer with Batch size 32

Model	Epoch	F1-score (%)	mAP@0.50 (%)	mAP@0.50:0.95 (%)	Time to train (h)	GPU
YOLOv5n	50	87.00	91.40	66.70	00:41	T4
	100	89.00	92.50	68.50	01:20	T4
	200	88.00	90.50	67.80	02:43	T4
YOLOv5m	50	89.00	92.50	70.20	01:24	T4
	100	88.00	93.10	71.50	02:40	T4
	200	89.00	92.60	71.80	01:46	V100
YOLOv6Lite-L	50	83.60	91.40	71.60	01:31	T4
	100	82.30	90.50	69.70	03:10	T4
	200	82.90	91.40	71.10	02:15	V100
YOLOv6n	50	83.50	89.90	67.10	01:56	T4
	100	83.40	90.30	66.70	03:31	T4
	200	84.70	91.20	68.80	01:58	V100
YOLOv7-tiny	50	80.00	87.30	58.90	01:00	T4
	100	85.00	91.40	65.00	02:00	T4
	200	86.00	90.80	66.20	02:00	V100
YOLOv8n	50	88.00	92.70	73.40	01:10	T4
	100	86.00	92.10	72.50	02:10	T4
	200	85.00	90.10	71.40	01:00	V100
YOLOv8m	50	89.00	**94.00**	**75.60**	01:26	T4
	100	87.00	92.00	72.80	03:00	T4
	200	**90.00**	**94.00**	74.20	01:58	V100

Of the various models evaluated, it was obtained that YOLOv8m in 50 epochs, using the SGD optimizer and batch size of 32, is the most optimal, obtaining high results of mAP@0.50 with 94.80% and mAP@0.50:0.95 with 76.60%, also an F1 of 88.00%. In addition, the selected model has a training time of 1:28h using a T4 GPU, making it ideal. For that reason, in Figure 2(a), the confusion matrix of the selected model is shown, visualizing most of the classes above 90.00% of correct answers. However, in the electric motorcycles class, there is a remarkable decrease, having only 61.00% of hits, there were too many false positives (44) as shown in Figure 2(b); this is due to similarities of some types of electric motorcycles with the electric scooters class as shown in Figures 3(a) and 3(b): Shape of the base of the vehicle, small tires and height of the rudder, this generates confusions in the trained model, being reflected in 38.00% of electric motorcycles misclassified as electric scooters. In addition, new elements have been detected (last column of the matrix), which could be erroneous detections, as in Figure 4(a), or objects that were not labeled in the data but were detected by the model, as in Figure 4(b). On the other hand, some objects were labeled but not detected (last row of the matrix) by the model, as in Figure 4(c), being only a minority.



(a)



(b)

Figure 2. Confusion matrix (a) normalized and (b) standard

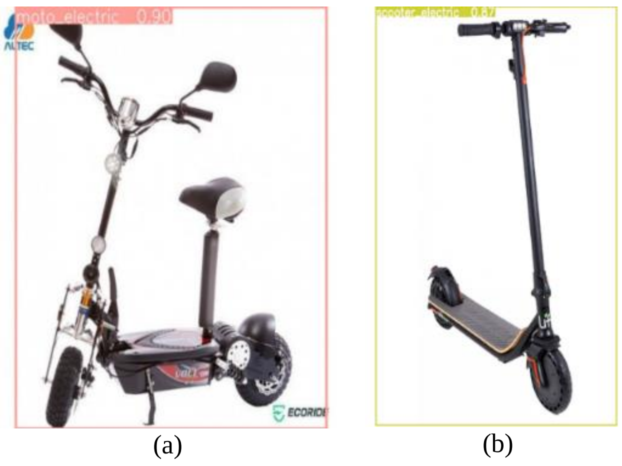
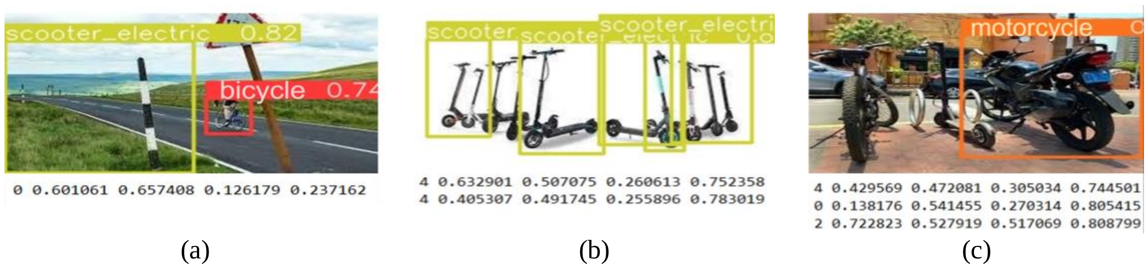


Figure 3. Similarity of characteristics between (a) an electric motorcycle and (b) an electric scooter



```
0 0.601061 0.657408 0.126179 0.237162
```

```
4 0.632901 0.507075 0.260613 0.752358
4 0.405307 0.491745 0.255896 0.783019
```

```
4 0.429569 0.472081 0.305034 0.744501
0 0.138176 0.541455 0.270314 0.805415
2 0.722823 0.527919 0.517069 0.808799
```

Figure 4. Elements detected (a) wrong, (b) new correctly, and (c) not detected

When comparing the investigation with others, such as [40] which also uses YOLOv8 (no model indicated) to detect 03 classes: traffic lights, motorcycles, and vehicles, with a database of 1000 images that were then introduced to the Roboflow page to increase the amount and diversify their data, obtaining about 15000 labels among their classes. They concluded that his model obtained an F1-score of 71.00% and a mAP@0.50 of 71.50%, but this is not correct because it takes values from graphs obtained from training and validation that are referential; the model may have learned or memorized the images designated for this process. The correct thing to do to validate the trained model is to take a group of images that it has not seen, as in the investigation performed, taking 964 images (1244 labels) designated for testing where the best model, in this case YOLOv8m, achieved an F1-score of 88.00% and a mAP@0.50 of 94.80%, demonstrating that the database collected is quite solid without using data augmentation techniques. Moreover, in [41] they made use of YOLOv7-tiny for the detection of 04 classes with a database of 13000 images, obtaining a mAP@0.50 of 91% with 300 epochs. In contrast, a database of 4852 images was used in the present investigation, obtaining a mAP@0.50 of 93.30% in 100 epochs with the same model. This demonstrates that the database is not about quantity but about having a robust set under different conditions and scenarios. In addition, using a virtual environment such as Google Colab is of great benefit to accelerate the training process because it gives you powerful tools when you do not have enough computational resources; training 300 epochs with a GTX3060ti took about 8.715 h while, in 100 epochs using the free virtual environment took about 2 h.

4. CONCLUSION

It is concluded from the ten models evaluated in various configurations, from the YOLOv4 version to the latest YOLOv8 version, that the YOLOv8m model in 50 epochs, using a batch size of 32 and SGD optimizer obtained the best mAP@0:50 and mAP@0.50:0.95 results, with 94.80% and 76.60% respectively, and an F1 of 88.00% with a training time of 1:28h using an Nvidia T4 GPU. Furthermore, it is inferred from all the results that the YOLOv8 version, just released in 2023, still under development and being tested by the community, outperforms the others in detection, also demonstrating that the database was good in the trained models. In future improvements, the database could be expanded and balanced to better define and divide the types of electric motorcycles and electric scooters in the market so that both classes could be correctly classified. Also, since we took light models of each YOLO version, the academic community is open to evaluating heavier models of each version.

REFERENCES

[1] F. B. -Maya, A. Calatayud, and V. González Mejía, “Estimating the effect of road congestion on air quality in Latin America,” *Transportation Research Part D: Transport and Environment*, vol. 113, Dec. 2022, doi: 10.1016/j.trd.2022.103510.

[2] G. Liu *et al.*, “Smart traffic monitoring system using computer vision and edge computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12027–12038, Aug. 2022, doi: 10.1109/TITS.2021.3109481.

[3] D. Chakraborty, S. Mohan, D. Dutta, and C. S. Jha, “Assessment of traffic congestion with high-resolution remote sensing data and deep convolution neural network,” *Geocarto International*, vol. 37, no. 23, pp. 6808–6825, Dec. 2022, doi: 10.1080/10106049.2021.1948110.

[4] J. Trivedi, M. S. Devi, and D. Dhara, “Vehicle classification using the convolution neuronal network approach,” *Scientific Journal of Silesian University of Technology. Series Transport*, vol. 112, pp. 201–209, Sep. 2021, doi: 10.20858/sjsutst.2021.112.7.16.

[5] F. Rofii, G. Priyandoko, Istiadi, and M. I. Fanani, “Modeling of convolutional neural networks for detection and classification of three vehicle classes,” *Journal of Physics: Conference Series*, vol. 1908, no. 1, Jun. 2021, doi: 10.1088/1742-6596/1908/1/012018.

[6] T. Singh, V. Rajput, Satakshi, U. Prasad, and M. Kumar, “Real-time traffic light violations using distributed streaming,” *The Journal of Supercomputing*, vol. 79, no. 7, pp. 7533–7559, May 2023, doi: 10.1007/s11227-022-04977-4.

[7] A. Mulyanto, W. Jatmiko, P. Mursanto, P. Prasetyawan, and R. I. Borman, “A new indonesian traffic obstacle dataset and performance evaluation of YOLOv4 for ADAS,” *Journal of ICT Research and Applications*, vol. 14, no. 3, pp. 286–298, Mar. 2021, doi: 10.5614/itbj.ict.res.appl.2021.14.3.6.

[8] K. P. H. and R. B. Venkatapur, “Deep learning technique for object detection from panoramic video frames,” *International Journal of Computer Theory and Engineering*, vol. 14, no. 1, pp. 20–26, 2022, doi: 10.7763/IJCTE.2022.V14.1306.

[9] M. Knura, F. Kluger, M. Zahtila, J. Schiewe, B. Rosenhahn, and D. Burghardt, “Using object detection on social media images for urban bicycle infrastructure planning: a case study of dresden,” *ISPRS International Journal of Geo-Information*, vol. 10, no. 11, Oct. 2021, doi: 10.3390/ijgi10110733.

[10] J. E. Espinosa, S. A. Velastin, and J. W. Branch, “Detection of motorcycles in urban traffic using video analysis: a review,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6115–6130, Oct. 2021, doi: 10.1109/TITS.2020.2997084.

[11] B. Zhu, M. Zhang, Yusu, and X. Hu, “Research on vehicle classification method based on improved AlexNet,” *Journal of Physics: Conference Series*, vol. 1955, no. 1, Jun. 2021, doi: 10.1088/1742-6596/1955/1/012060.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[13] I. V. Fanthony, Z. Husin, H. Hikmarika, S. Dwijayanti, and B. Y. Suprapto, “YOLO algorithm-based surrounding object identification on autonomous electric vehicle,” in *2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, IEEE, Oct. 2021, pp. 151–156, doi: 10.23919/EECSI53397.2021.9624275.

[14] A. Dairi, F. Harrou, Y. Sun, and M. Senouci, “Obstacle detection for intelligent transportation systems using deep stacked autoencoder and k-nearest neighbor scheme,” *IEEE Sensors Journal*, vol. 18, no. 12, pp. 5122–5132, Jun. 2018, doi: 10.1109/JSEN.2018.2831082.

[15] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 2, pp. 386–397, Feb. 2020, doi: 10.1109/TPAMI.2018.2844175.

[16] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A review of YOLO algorithm developments,” *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022, doi: 10.1016/j.procs.2022.01.135.

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: optimal speed and accuracy of object detection,” *arXiv-Computer Science*, pp. 1-17, Apr. 2020.

[18] N. N. F. Giron, R. K. C. Billones, A. M. Fillone, J. R. D. Rosario, A. A. Bandala, and E. P. Dadios, “Classification between pedestrians and motorcycles using FasterRCNN inception and SSD MobileNetv2,” in *2020 IEEE 12th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, IEEE, Dec. 2020, pp. 1–6, doi: 10.1109/HNICEM51456.2020.9400113.

[19] U. Kamal, T. I. Tonmoy, S. Das, and Md. K. Hasan, “Automatic traffic sign detection and recognition using SegU-Net and a modified tversky loss function with L1-constraint,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1467–1479, Apr. 2020, doi: 10.1109/TITS.2019.2911727.

[20] L. Chen *et al.*, “Deep neural network based vehicle and pedestrian detection for autonomous driving: a survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3234–3246, Jun. 2021, doi: 10.1109/TITS.2020.2993926.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: inverted residuals and linear bottlenecks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[22] P. Nagrath, R. Jain, A. Madan, R. Arora, P. Kataria, and J. Hemanth, “SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2,” *Sustainable Cities and Society*, vol. 66, Mar. 2021, doi: 10.1016/j.scs.2020.102692.

[23] T. P. Dang, N. T. Tran, V. H. To, and M. K. T. Thi, “Improved YOLOv5 for real-time traffic signs recognition in bad weather conditions,” *The Journal of Supercomputing*, vol. 79, no. 10, pp. 10706–10724, Jul. 2023, doi: 10.1007/s11227-023-05097-3.

[24] Z. Ren, H. Zhang, and Z. Li, “Improved YOLOv5 network for real-time object detection in vehicle-mounted camera capture scenarios,” *Sensors*, vol. 23, no. 10, May 2023, doi: 10.3390/s23104589.

[25] L. Shao, H. Wu, C. Li, and J. Li, “A vehicle recognition model based on improved YOLOv5,” *Electronics*, vol. 12, no. 6, Mar. 2023, doi: 10.3390/electronics12061323.

[26] Q. Zhao, W. Ma, C. Zheng, and L. Li, “Exploration of vehicle target detection method based on lightweight YOLOv5 fusion background modeling,” *Applied Sciences*, vol. 13, no. 7, Mar. 2023, doi: 10.3390/app13074088.

[27] C. J. M. Dequito *et al.*, “Vision-based bicycle and motorcycle detection using a YOLO-based Network,” *Journal of Physics: Conference Series*, vol. 1922, no. 1, May 2021, doi: 10.1088/1742-6596/1922/1/012003.

[28] J. Redmon and A. Farhadi, “YOLOv3: an incremental improvement,” *arXiv-Computer Science*, pp. 1-6, Apr. 2018.

[29] Y. Miao, F. Liu, T. Hou, L. Liu, and Y. Liu, “A Nighttime Vehicle Detection Method Based on YOLO v3,” in *2020 Chinese Automation Congress (CAC)*, IEEE, Nov. 2020, pp. 6617–6621, doi: 10.1109/CAC51589.2020.9326819.

[30] E. P. -Peredo, "Deep learning for the classification of cassava leaf diseases in unbalanced field data set," *Advanced Network Technologies and Intelligent Computing*, 2023, pp. 101–114, doi: 10.1007/978-3-031-28183-9_8.
[31] A. B. -Cerquín, J. T. Guevara, E. A. P. Peredo, and J. Palomino, "Peruvian sign language translator for people with hearing and/or communication disabilities using a convolutional neural network," in *21st LACCEI International Multi-Conference for Engineering, Education, and Technology*, Latin American and Caribbean Consortium of Engineering Institutions, 2023, doi: 10.18687/LACCEI2023.1.1.1403.
[32] D. A. C. -Laurentt and E. A. P. -Peredo, "Histopathological image classification using convolutional neural networks for detection of metastatic breast cancer in lymph nodes," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 20, no. 2, pp. 31–45, Feb. 2024, doi: 10.3991/ijoe.v20i02.46789.
[33] J.-D. Wu, B.-Y. Chen, W.-J. Shyr, and F.-Y. Shih, "Vehicle classification and counting system using YOLO object detection technology," *Traitement du Signal*, vol. 38, no. 4, pp. 1087–1093, Aug. 2021, doi: 10.18280/ts.380419.
[34] M. A. Berwo, Z. Wang, Y. Fang, J. Mahmood, and N. Yang, "Off-road quad-bike detection using CNN models," *Journal of Physics: Conference Series*, vol. 2356, no. 1, Oct. 2022, doi: 10.1088/1742-6596/2356/1/012026.
[35] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: scaling cross stage partial network," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 13024–13033, doi: 10.1109/CVPR46437.2021.01283.
[36] H. Gomes, N. Redinha, N. Lavado, and M. Mendes, "Counting people and bicycles in real time using YOLO on Jetson nano," *Energies*, vol. 15, no. 23, Nov. 2022, doi: 10.3390/en15238816.
[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
[38] C. Li *et al.*, "YOLOv6: a single-stage object detection framework for industrial applications," *arXiv-Computer Science*, pp. 1-17, Sep. 2022.
[39] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2022, pp. 7464-7475, doi: 10.1109/CVPR52729.2023.00721.
[40] M. S. Beg, M. Y. Ismail, and M. S. U. Miah, "Evaluating the performance of a visual support system for driving assistance using a deep learning algorithm," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 34, no. 1, pp. 38–50, Nov. 2023, doi: 10.37934/araset.34.1.3850.
[41] I. Elmanaa, M. A. Sabri, Y. Abouch, and A. Aarab, "efficient roundabout supervision: real-time vehicle detection and tracking on nvidia jetson nano," *Applied Sciences*, vol. 13, no. 13, Jun. 2023, doi: 10.3390/app13137416

BIOGRAPHIES OF AUTHORS

Joshue Garcia-Pajuelo Bachelor in Electronic Engineering of the Faculty of Systems Engineering and Electronics (FISE) of the Universidad Tecnológica del Perú in Lima with code 1310406. He is particularly interested in artificial intelligence, machine learning, and deep learning. He can be contacted at email: 1310406@utp.edu.pe.

Ernesto Paiva-Peredo received the title of Electrical Mechanical Engineer from the Universidad de Piura, Peru, in 2013. He has completed a master's degree in electrical mechanical engineering with a mention in Autom00ation and Optimization at the Universidad de Piura funded by CONCYTEC 2016. He was a research assistant at the Department of Technology and Innovation (DTI) - SUPSI. Now, he is a Professor-Researcher at Universidad Tecnológica del Perú. He can be contacted at email: epaiva@utp.edu.pe.